

*Curs 3*

*Colectarea cerințelor*

*Curs bazat pe B. Bruegge and A.H. Dutoit*

*"Object-Oriented Software Engineering using UML, Patterns, and Java"*

## Sumar Curs 3

---

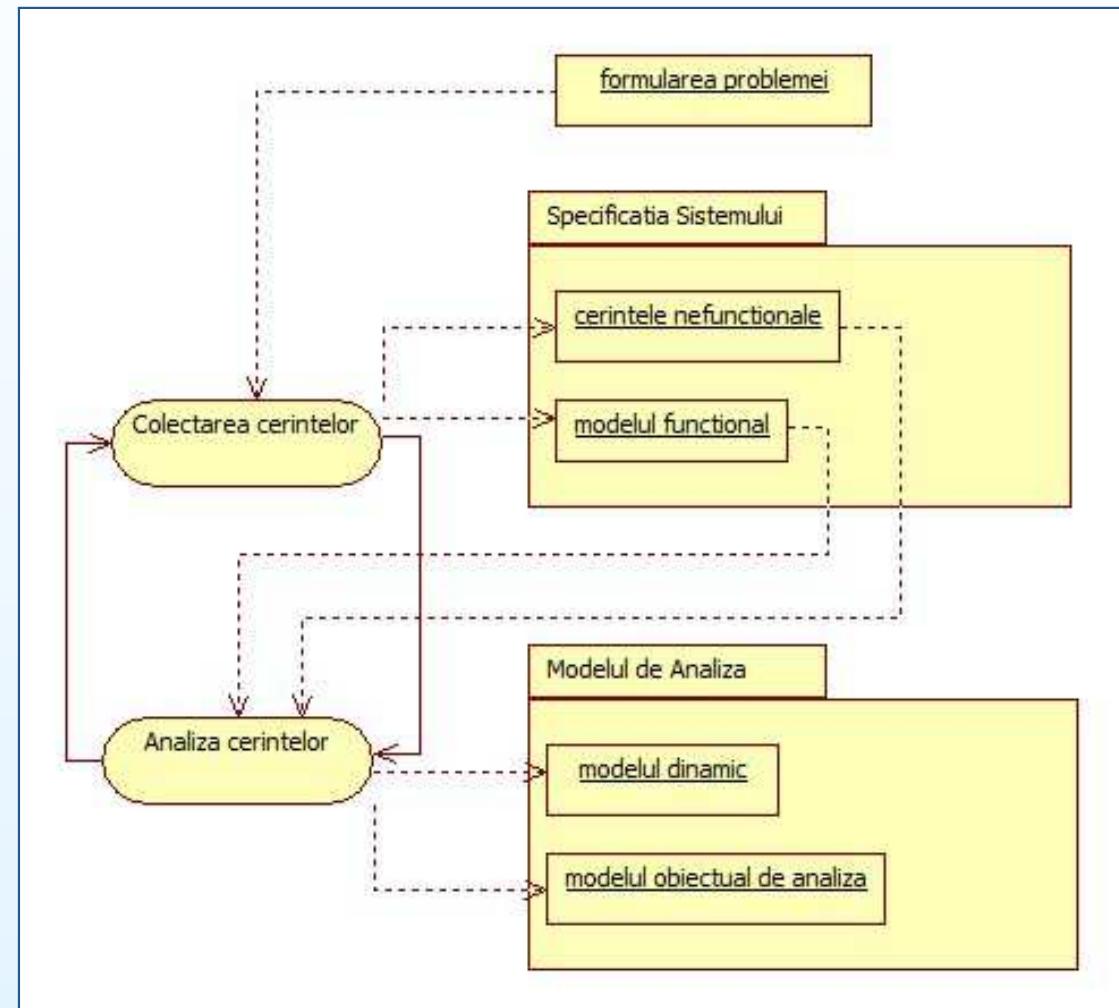
- Cerințe. Ingineria cerințelor
- Colectarea cerințelor - concepte
- Colectarea cerințelor - activități tehnice
- Colectarea cerințelor - management

# Cerințe. Ingineria cerințelor

- O cerință (eng. *requirement*) reprezintă un element de funcționalitate pe care sistemul trebuie să îl ofere sau o constrângere pe care trebuie să o îndeplinească
- *Ingineria cerințelor* (eng. *requirements engineering*) este un subdomeniu al ingineriei softului, având drept scop definirea cerințelor sistemelor soft ce urmează a fi construite
- Activități ale ingineriei cerințelor
  - Colectarea cerințelor (eng. *requirements elicitation*)  $\implies$  *specificația sistemului* (contract între client și dezvoltatori)
  - Analiza cerințelor (eng. *analysis*)  $\implies$  *modelul de analiză*
- Atât specificația sistemului, cât și modelul de analiză reflectă aceeași informație, ele diferă însă prin rolul lor și notația utilizată
  - Specificația sistemului este exprimată în limbaj natural și servește ca și instrument de comunicare cu clienții/utilizatorii
  - Modelul de analiză este exprimat într-o notație semiformală sau formală și servește ca și instrument de comunicare între dezvoltatori

## Cerințe. Ingineria cerințelor (cont.)

- Activități și produse ale ingineriei cerințelor (diagramă UML de activități)
  - Activitățile ingineriei cerințelor se focusează exclusiv pe aspectele externe ale sistemului (perspectiva utilizatorilor externi asupra sistemului)
  - Modelele lor reprezentând aceleași aspecte, cele două activități se desfășoară concurrent și iterativ



# Colectarea cerințelor

- Colectarea cerințelor necesită colaborarea între grupuri de participanți cu background diferit (cliienți/utilizatori - experți în domeniul problemei vs. dezvoltatori - experți în dezvoltare de soft)
  - Corectarea erorilor de comunicare introduse acum (rezultând în specificarea greșită a unor funcționalități, funcționalități lipsă, probleme la nivelul interfețelor utilizator) în etapele ulterioare ale dezvoltării este deosebit de costisitoare
- Metodele de colectare a cerințelor au drept obiectiv perfecționarea comunicării între aceste grupuri
- *Colectarea cerințelor bazată pe scenarii* (eng. *scenario-based requirements elicitation*)
  - *Scenariu* = descriere a unui exemplu concret de utilizare a sistemului, în termenii unei secvențe de interacțiuni între utilizator și sistem
  - *Caz de utilizare* = abstractizare ce descrie elementele comune ale unei clase de scenarii

## Colectarea cerințelor bazată pe scenarii

- Dezvoltatorii colectează cerințele prin observarea și chestionarea utilizatorilor în mediul lor
- Se realizează două tipuri de scenarii
  - Inițial - scenarii descriind modul curent de desfășurare a proceselor de lucru în mediul utilizatorilor
  - Ulterior - scenarii prescriptive, ilustrând funcționalitatea ce urmează a fi oferită de noul sistem
- Scenariile realizate sunt validate de către clienți și utilizatori prin inspectare și prin manipularea unor prototipuri ale interfeței grafice cu utilizatorul, oferite de către dezvoltatori
- Pe măsură ce definiția sistemului se stabilizează, dezvoltatorii și clientul convin asupra unei specificații a sistemului constând în descrierea cerințelor funcționale, a celor nefuncționale, a cazurilor de utilizare și a scenariilor aferente

# Activități ale colectării bazate pe scenarii

---

- *Identificarea actorilor*
  - Dezvoltatorii identifică diferitele tipuri de utilizatori pe care îi va suporta sistemul
- *Identificarea scenariilor*
  - Dezvoltatorii observă utilizatorii în mediul lor și descriu un set de scenarii detaliate aferente funcționalităților tipice oferite de noul sistem
  - Scenariile sunt utilizate pentru comunicarea cu utilizatorii și aprofundarea înțelegерii domeniului problemei de către dezvoltatori
- *Identificarea cazurilor de utilizare*
  - Ulterior stabilizării scenariilor, dezvoltatorii derivă din acestea cazurile de utilizare ce definesc viitorul sistem
- *Rafinarea cazurilor de utilizare*
  - Dezvoltatorii verifică completitudinea specificării cerințelor prin detalierea fiecărui caz de utilizare și descrierea comportamentului sistemului în situații de excepție

## Activități ale colectării bazate pe scenarii (cont.)

---

- *Identificarea relațiilor dintre cazurile de utilizare*
  - Dezvoltatorii identifică dependențele dintre cazurile de utilizare și factorizează comportamentele comune
  - Această activitate permite verificarea consistenței specificației sistemului
- *Identificarea cerințelor nefuncționale*
  - Dezvoltatorii, clienții și utilizatorii convin asupra constrângerilor legate de performanțele sistemului, resursele utilizate, securitatea și calitatea sa, modalitatea de documentare, etc.

## Colectarea cerințelor - concepte

---

- Cerințe funcționale
- Cerințe nefuncționale
- Completitudine
- Consistență
- Claritate
- Corectitudine
- Realism
- Verificabilitate
- Trasabilitate
- Inginerie Greenfield
- Re-inginerie
- Ingineria interfețelor

## Cerințe funcționale

- *Cerințele funcționale* (eng. *functional requirements*) descriu interacțiunile dintre sistem și mediul acestuia, independent de implementare
  - Mediul = utilizatorii + alte sisteme cu care sistemul în cauză interacționează
- Ex.: Specificarea cerințelor funcționale ale sistemului *SatWatch* - ceas care se actualizează automat (fără intervenții externe)
  - *SatWatch* este un ceas de mână care afișează timpul pe baza locației curente. *SatWatch* utilizează sateliți GPS pentru a determina locația curentă și structuri interne pentru a asocia acelei locații un fus orar.  
*SatWatch* actualizează ora și data la schimbarea fusului orar sau a granițelor politice. Proprietarul ceasului nu trebuie să îl reseteze niciodată, ca urmare ceasul nu are butoane de control.  
*SatWatch* utilizează sateliți GPS pentru determinarea locației curente, ca urmare are aceleași limitări ca și restul dispozitivelor GPS (ex. incapacitatea de a determina locația la anumite momente din zi, în regiunile muntoase). În timpul perioadelor de black-out, *SatWatch* consideră că nu se schimbă fusurile orare sau regiunile politice. Imediat după ieșirea din intervalul de black-out, se realizează actualizările aferente.

## Cerințe funcționale (cont.)

*SatWatch* are un afisaj pe două linii, cea de sus indicând timpul (ora, minute, secunde, fus orar), iar cea de jos data (zi din săptămână, data, luna, an).

În cazul modificării granițelor politice, proprietarul poate actualiza softul ceasului prin utilizarea dispozitivului *WebifyWatch* (oferit împreună cu ceasul) și a unui calculator conectat la Internet.

- Cerințele funcționale anterioare surprind doar interacțiunile dintre sistemul *SatWatch* și mediul extern (proprietarul, sateliții GPS, dispozitivul *WebifyWatch*), fără a referi detalii de implementare (procesor, limbaj, tehnologie, etc.)

# Cerințe nefuncționale

- *Cerințele nefuncționale* (eng. *nonfunctional requirements*) descriu diverse tipuri de constrângeri impuse asupra sistemului, ortogonale pe aspectele funcționale
- Categorii de cerințe nefuncționale (*de calitate*), conform modelului FURPS+ ([Grady 1992], [IEEE Std. 610.12-1990])
  - *Utilizabilitate* (eng. *usability*)
    - Denotă ușurința cu care un utilizator poate învăța să opereze, să pregătească intrări sau să interpreteze ieșiri ale unui sistem sau ale unei componente
    - Ex. de cerințe privind utilizabilitatea: diferite conveții adoptate la nivelul interfețelor grafice (șabloane de structurare, scheme de culori, logo-uri, fonturi), help online, ghid de utilizare detaliat
  - *Performanță* (eng. *performance*) - referă atribute cuantificabile, precum:
    - *timpul de răspuns* (rapiditatea reacției sistemului la inputul utilizatorului)
    - *puterea de calcul* (volumul de calcule efectuate într-un interval de timp)
    - *acuratețea rezultatelor*
    - *disponibilitatea* (măsura în care un sistem este operațional și accesibil atunci când se dorește utilizarea lui)

## Cerințe nefuncționale (cont.)

- *Fiabilitate* (eng. *reliability*)
  - Reprezintă abilitatea sistemului sau componentei de a îndeplini funcțiile cerute în condițiile stabilită, pentru o anumită perioadă de timp
  - Ex. de cerințe privind fiabilitatea: un timp mediu precizat între eșecuri, abilitatea de a face față unor atacuri de securitate, etc.
  - Mai nou referintă cu termenul de *dependability* și incluzând, în afara fiabilității propriu-zise și *robustetea* (eng. *robustness*) - gradul în care un sistem sau o componentă poate funcționa corect în prezența unor intrări invalide sau a unor condiții excepționale și *siguranța* (eng. *safety*) - o măsură a absenței unor consecințe catastrofale în mediul
- *Suportabilitate* (eng. *suportability*)
  - Denotă ușurința schimbării sistemului după instalare, incluzând *adaptabilitatea* (eng. *adaptability*) - abilitatea de a modifica sistemul pentru a opera cu noi concepte din domeniul problemei, *mentenabilitatea* (eng. *maintainability*) - abilitatea de a schimba sistemul pentru a opera cu noi tehnologii sau a repara defecte și *internationalizarea* (eng. *internationalization*) - abilitatea de a schimba sistemul pentru a opera cu limbi/unități de măsură/formate numerice străine

## Cerințe nefuncționale (cont.)

- Categorii adiționale de cerințe calificate drept nefuncționale în modelul FURPS+ (*pseudo-cerințe sau constrângeri*)
  - *Cerințe de implementare* (eng. *implementation requirements*)
    - Sunt constrângeri ce vizează utilizarea unei anumite platforme hardware, a unui anumit limbaj de programare sau a anumitor instrumente
  - *Cerințe privind interfața* (eng. *interface requirements*)
    - Sunt constrângeri impuse de alte sisteme cu care sistemul în cauză interfețează
  - *Cerințe privind modul de operare* (eng. *operations requirements*)
    - Sunt constrângeri privind administrarea și gestiunea sistemului în mediul său de operare
  - *Cerințe privind instalarea* (eng. *packaging requirements*)
    - Sunt constrângeri legate de livrarea sistemului, cum ar fi suportul folosit pentru instalare
  - *Cerințe legale* (eng. *legal requirements*)
    - Sunt constrângeri legate de licențe, legi, certificări
    - Ex.: cerințe privind obligativitatea oferirii accesului la un anumit soft persoanelor cu dizabilități

## Cerințe nefuncționale (cont.)

- Ex.: cerințe de calitate pentru *SatWatch*
  - Orice utilizator care știe să citească un ceas digital și cunoaște abrevierile legate de fusurile orare trebuie să poată folosi sistemul fără manual de utilizare (utilizabilitate)
  - *SatWatch* trebuie să afișeze fusul orar corect în maxim 5 minute de la ieșirea dintr-o perioadă de black-out (performanță)
  - Deoarece *SatWatch* nu dispune de butoane, nu trebuie să apară nici o eroare care să necesite resetarea acestuia (fiabilitate)
  - *SatWatch* trebuie să admită upgrade-uri prin intermediul interfeței seriale *WebifyWatch* (suportabilitate)
- Ex.: constrângeri pentru *SatWatch*
  - Softul *SatWatch* va fi scris integral în Java, pentru conformanță cu standardele companiei (implementare)
  - *SatWatch* se conformează interfețelor fizice, electrice și soft definite de *WebifyWatch 2.0* (interfață)

# Completitudine, consistență, claritate și corectitudine

---

- Validarea continuă a cerințelor reprezintă un imperativ în procesul de dezvoltare
- Validarea cerințelor presupune verificarea completitudinii, consistenței, clarității și corectitudinii acestora
  - *Completitudine*
    - Specificarea cerințelor se consideră a fi completă dacă surprinde toate aspectele de interes pentru clienți/utilizatori (au fost descrise toate scenariile posibile, inclusiv cele de excepție)
  - *Consistență*
    - Specificarea cerințelor se consideră a fi consistentă dacă nu există două cerințe care să se contrazică reciproc
  - *Claritate*
    - Specificarea cerințelor se consideră a fi neambiguă dacă o aceeași cerință nu admite două interpretări distincte (definește un singur sistem)
  - *Corectitudine*
    - Specificarea cerințelor se consideră a fi corectă dacă reprezintă fidel interesele clientului și ale dezvoltatorilor, fără a include elemente nedorite
- Corectitudinea și completitudinea sunt greu de apreciat/certificat, mai ales anterior existenței sistemului

# Realism, verificabilitate și trasabilitate

- Specificarea cerințelor se consideră a fi *realistă* dacă sistemul poate fi implementat cu constrângerile impuse
- Specificarea cerințelor se consideră a fi *verificabilă* dacă ulterior dezvoltării sistemului pot fi proiectate teste care să dovedească conformanța acestuia cu specificația
  - Ex.: cerințe greu/imposibil de verificat
    - Sistemul *SatWatch* trebuie să aibă un interval mediu între eșecuri de 100 de ani
    - Produsul trebuie să aibă o interfață utilizator bună
    - Produsul trebuie să fie fără erori
- O specificare a cerințelor are proprietatea de *trasabilitate* dacă fiecare cerință poate fi urmărită, de-a lungul procesului de dezvoltare, până la funcțiile sistem care o implementează și reciproc
  - Trasabilitatea cerințelor e o constrângere critică pentru dezvoltarea testelor și analiza impactului schimbărilor asupra sistemului

# Trasabilitatea cerințelor

- *Trasabilitatea* (eng. *traceability*) reprezintă abilitatea de a urmări evoluția unei cerințe
  - Presupune urmărirea cerinței de la origine (cine a introdus-o? cărei nevoi client corespunde?) până la nivelul acelor componente ale sistemului pe care le afectează (ce componente implementează cerința? ce cazuri de test verifică îndeplinirea sa?)
  - Trasabilitatea permite dezvoltatorilor să motiveze completitudinea sistemului, testerilor să justifice conformanța acestuia cu cerințele, proiectanților să înregistreze argumentele din spatele deciziilor luate și echipelor de întreținere să evaluateze efectul schimbărilor
- Trasabilitatea poate fi urmărită prin întreținerea de referințe între documente, modele și cod
  - Fiecare element individual (cerință, componentă, clasă, operație, caz de test) primește un identificator unic
  - O dependență este apoi documentată prin intermediul unei referințe conținând identificatorii componentelor sursă și destinație

# Inginerie Greenfield, re-inginerie, ingineria interfețelor

---

- Caracteristicile activității de colectare a cerințelor depind de sursa acestora
  - *Ingineria Greenfield*
    - Procesul de dezvoltare începe de la 0, nu există un sistem anterior
    - Cerințele sunt furnizate doar de client și utilizatori
  - *Re-inginerie*
    - Reproiectarea și reimplementarea unui sistem existent, ca urmare a unor modificări la nivelul proceselor de lucru sau a unor modificări tehnologice
    - Funcționalitatea sistemului poate fi eventual extinsă, însă scopul său rămâne același; majoritatea cerințelor sunt extrase din vechiul sistem
  - *Ingineria interfețelor*
    - Reproiectarea și reimplementarea doar a interfeței unui sistem existent
- În cazul ingineriei Greenfield și a reingineriei, dezvoltatorii trebuie să acumuleze cât mai multe cunoștințe relativ la domeniul problemei
  - Surse: descrieri ale proceselor de lucru, documentație distribuită noilor angajați, manuale ale vechiului sistem, note ale utilizatorilor, interviuri cu clienții și utilizatorii

## Colectarea cerințelor - activități tehnice

---

- Identificarea actorilor
- Identificarea scenariilor
- Identificarea cazurilor de utilizare
- Rafinarea cazurilor de utilizare
- Identificarea relațiilor între cazurile de utilizare
- Identificarea cerințelor nefuncționale

## Identificarea actorilor

- Permite identificarea frontierei sistemului și a perspectivelor din care acesta trebuie abordat
- Întrebări utile pentru identificarea actorilor
  - Care sunt grupurile de utilizatori a căror activitate este susținută de sistem?
  - Care sunt grupurile de utilizatori care execută funcțiile principale ale sistemului?
  - Care sunt grupurile de utilizatori care execută funcții secundare, precum întreținere sau administrare?
  - Care sunt echipamentele hardware și sistemele software externe cu care sistemul curent va interacționa?
- Întrebările anterioare conduc la identificarea unei liste de entități ce urmează a fi rafinată, în general, într-un număr mai mic de actori, diferenți din perspectiva utilizării sistemului
  - Se unifică entitățile care utilizează aceleași interfețe
  - Se elimină entitățile care, cel mai probabil, nu vor interacționa în mod direct cu sistemul

## Identificarea scenariilor

- În colectarea cerințelor, dezvoltatorii și utilizatorii concep și rafinează o serie de scenarii, pentru a dobândi un nivel comun de înțelegere relativ la funcționalitatea sistemului
- Întrebări utile pentru identificarea scenariilor
  - Care sunt sarcinile pe care utilizatorul dorește să le execute sistemul?
  - Care sunt informațiile pe care le poate accesa actorul? Cine creează aceste date? Pot fi ele modificate sau șterse? De către cine?
  - Care sunt modificările externe despre care actorul trebuie să informeze sistemul? Cât de des? Când?
  - Care sunt evenimentele despre care sistemul trebuie să informeze actorul? Cu ce periodicitate?
- Pentru a răspunde întrebărilor anterioare, dezvoltatorii consultă diverse documente cu informații privind domeniul problemei
  - manuale ale unor sisteme anterioare, manuale procedurale, standarde ale companiei, note și documente elaborate de utilizatori, interviuri cu clienții și utilizatorii

## Identificarea scenariilor (cont.)

- Ex.: Scenariul *depozitIncendiat* al cazului de utilizare *RaporteazăUrgență* al SGA

<b>Nume</b>	<u>depozitIncendiat</u>
<b>Instanțe</b>	<u>bob, alice : OfițerTeren</u>
<b>actori</b>	<u>john : Dispecer</u>
<b>Flux de evenimente</b>	<ol style="list-style-type: none"><li>1. Trecând prin dreptul unui depozit, Bob simte miros de fum. Partenera sa, Alice, activează funcția <i>Raportează urgență</i> pe terminalul SGA.</li><li>2. Alice introduce adresa clădirii, o scurtă descriere a locației și un nivel de alertă. Zona fiind aglomerată, solicită o echipă de pompieri și mai multe de medici. Confirmă datele și așteaptă confirmarea dispecerului.</li><li>3. John, dispecerul, este alertat de un semnal sonor al stației sale de lucru. Verifică informațiile trimise de Alice și confirmă primirea lor. Alocă o echipă de pompieri și două de medici și ii trimitе lui Alice ora estimată a sosirii acestora.</li><li>5. Alice primește confirmarea și estimarea.</li></ol>

# Identificarea cazurilor de utilizare

- Ex.: Cazul de utilizare *RaporteazăUrgență* al SGA

<b>Nume</b>	<i>RaporteazăUrgență</i>
<b>Participanți</b>	Inițiat de <i>OfițerTeren</i> Comunică cu <i>Dispecerul</i>
<b>Flux de evenimente (scenariu normal)</b>	<ol style="list-style-type: none"><li>1. <i>Ofițerul</i> activează funcția <i>Raportează urgență</i> a terminalului.</li><li>2. Sistemul SGA afișează un formular <i>Ofițerului</i>.</li><li>3. <i>Ofițerul</i> completează formularul, inserând nivelul de alertă, tipul, locația și o scurtă descriere a situației. Poate propune și eventuale soluții la situația de urgență. După completare, <i>Ofițerul</i> trimitе formularul.</li><li>4. Sistemul primește formularul și notifică <i>Dispecerul</i>.</li><li>5. <i>Dispecerul</i> consultă informația primită și apelează cazul de utilizare <i>DeschideCazNou</i>. <i>Dispecerul</i> optează pentru una dintre soluțiile propuse și confirmă primirea formularului.</li><li>6. Sistemul afișează confirmarea și soluția aleasă <i>Ofițerului</i>.</li></ol>
<b>Condiții de intrare</b>	<i>Ofițerul</i> este logat în sistem.
<b>Condiții de ieșire</b>	<i>Ofițerul</i> a primit confirmarea de la <i>Dispecer</i> SAU o explicație privind motivul eșecului tranzacției.
<b>Cerințe de calitate</b>	Confirmarea <i>Dispecerului</i> ajunge în maxim 30 de sec. după trimitere.

## Identificarea cazurilor de utilizare (cont.)

---

- Ghid de descriere a cazurilor de utilizare
  - Numele cazurilor de utilizare trebuie să fie construcții verbale, care să indice scopul utilizatorului (ex. *RaporteazăUrgență*, *AlocăResurse*)
  - Numele actorilor trebuie să fie substantive (ex. *OfițerTeren*, *Dispecer*)
  - Frontiera sistemului trebuie să fie clară, distingându-se între acțiunile actorilor și cele ale sistemului
  - Evenimentele care definesc cazul de utilizare trebuie formulate la modul activ
  - Relația de cauzalitate între două evenimente consecutive trebuie să fie clară
  - Fluxul normal al unui caz de utilizare trebuie să descrie o tranzacție completă (ex.: fluxul normal al cazului *RaporteazăUrgență* descrie toți pașii, de la inițierea raportului și până la primirea confirmării finale)
  - Excepțiile/alternativele se descriu separat
  - Descrierea unui caz de utilizare trebuie să evite referirea directă a componentelor interfeței grafice cu utilizatorul
  - Descrierea unui caz de utilizare nu trebuie să depășească două-trei pagini. În caz contrar, cazul se descompune, folosind relațiile între cazuri de utilizare

# Rafinarea cazurilor de utilizare

---

- Focusul acestei activități îl constituie completitudinea și corectitudinea specificării cerințelor
  - Dezvoltatorii identifică funcționalități neacoperite de scenariile descrise (cazuri rare sau excepții) și le documentează prin rafinarea cazurilor de utilizare existente sau introducerea unor noi cazuri
  - Spre deosebire de identificarea inițială a actorilor și cazurilor de utilizare, focusată pe definirea frontierei sistemului, etapa de rafinare introduce detalii suplimentare privind funcționalitățile oferite de sistem și constrângerile asociate
- Sunt detaliate următoarele aspecte
  - Informația manipulată de sistem
  - Interacțiunile dintre actori și sistem
  - Drepturile de acces (ce cazuri de utilizare poate invoca fiecare actor)
  - Cazurile de excepție (sunt identificate și specificate)
  - Funcționalitatea comună (este factorizată)

## Rafinarea cazurilor de utilizare (cont.)

- Ex.: Rafinarea cazului de utilizare *RaporteazăUrgență*

<b>Nume</b>	<i>RaporteazăUrgență</i>
<b>Participanți</b>	<i>Inițiat de OfițerTeren Comunică cu Dispecerul</i>
<b>Flux de evenimente</b>	<ol style="list-style-type: none"><li>1. Ofițerul activează funcția <i>Raportează urgență</i> a terminalului.</li><li>2. Sistemul SGA afișează un formular <i>Ofițerului</i>. <i>Formularul include componente privind nivelul de alertă, tipul urgenței (general, incendiu, accident auto), locația, descrierea și resursele solicitate.</i></li><li>3. Ofițerul completează formularul, <i>inserând cel puțin nivelul de alertă și descrierea situației</i>. El poate propune și eventuale soluții la situația de urgență și poate să solicite anumite resurse. După completare, <i>Ofițerul</i> trimite formularul.</li><li>4. Sistemul primește formularul și notifică <i>Dispecerul</i>.</li><li>5. <i>Dispecerul</i> verifică informația primită și creează un nou <i>Eveniment</i> în baza de date prin invocarea cazului de utilizare <i>DeschideCazNou</i>. <i>Toate informațiile din formularul primit sunt asociate automat evenimentului creat. Dispecerul alege un răspuns prin alocarea de resurse la eveniment (prin cazul de utilizare AlocăResurse) și confirmă primirea formularului printr-un scurt mesaj către ofițer.</i></li><li>6. Sistemul afișează confirmarea și răspunsul ales <i>Ofițerului</i>.</li></ol>

...

## Identificarea relațiilor

---

- Identificarea relațiilor dintre actori și cazurile de utilizare (comunicare, incluziune, extindere, generalizare) permite reducerea complexității modelului, eliminarea redundanțelor și a potențialelor inconsistențe
- Euristică privind folosirea relațiilor de incluziune și extindere
  - Utilizarea relației de incluziune pentru factorizarea comportamentului comun mai multor cazuri de utilizare
  - Utilizarea relației de extindere pentru modelarea comportamentului opțional, excepțional sau cu frecvență redusă de apariție
  - Utilizarea judicioasă a relațiilor, pentru a evita suprastructurarea modelului funcțional. Un singur caz de utilizare mai complex se poate dovedi a fi mai inteligibil decât un număr prea mare de cazuri mai simple

# Identificarea cerințelor nefuncționale

- Întrebări ce permit identificarea cerințelor nefuncționale aferente modelului FURPS+
  - Utilizabilitate
    - Care este nivelul de expertiză al utilizatorilor?
    - Ce standarde de interfețe sunt familiare utilizatorilor?
    - Ce documentație trebuie pusă la dispoziția utilizatorilor?
  - Fiabilitate (incluzând robustețe, siguranță și securitate)
    - Cât de fiabil/robust trebuie să fie sistemul?
    - Este repornirea sistemului o alternativă viabilă în cazul unui eșec?
    - Sunt admisibile pierderi de date? Ce volum?
    - Cum trebuie să gestioneze sistemul excepțiile?
    - Există cerințe privind siguranță?
    - Există cerințe privind securitatea?
  - Suportabilitate
    - Care sunt extinderile probabile ale sistemului?
    - Cine întreține sistemul?
    - Există probabilitatea de a porta sistemul pe alte platforme hardware/software?

## Identificarea cerințelor nefuncționale (cont.)

---

- Performanță
  - Există sarcini critice din punct de vedere al timpului?
  - Câți utilizatori concurenți trebuie să suporte sistemul?
  - Ce dimensiuni se estimează că va avea depozitul de date?
  - Care este cel mai slab timp de răspuns acceptat de utilizatori?
- Implementare
  - Există constrângeri privind platforma hardware?
  - Există constrângeri impuse echipei de testare?
  - Există constrângeri impuse echipei de întreținere?
- Interfață
  - Va interacționa sistemul cu alte sisteme?
  - Cum sunt exportate/importate datele la nivelul sistemului?
  - Există standarde utilizate de client care se aplică noului sistem?
- Instalare
  - Cine va instala sistemul?
  - Pe câte stații va fi sistemul instalat?
  - Există constrângeri de timp privind instalarea?

## Identificarea cerințelor nefuncționale (cont.)

---

- Operare
  - Cine gestionează sistemul după punerea în exploatare?
- Legal
  - Care este procedura de licențiere?
  - Există taxe rezultând din utilizarea anumitor componente/algoritmi?
  - Există penalizări în cazul căderii sistemului?

## **Colectarea cerințelor - management**

---

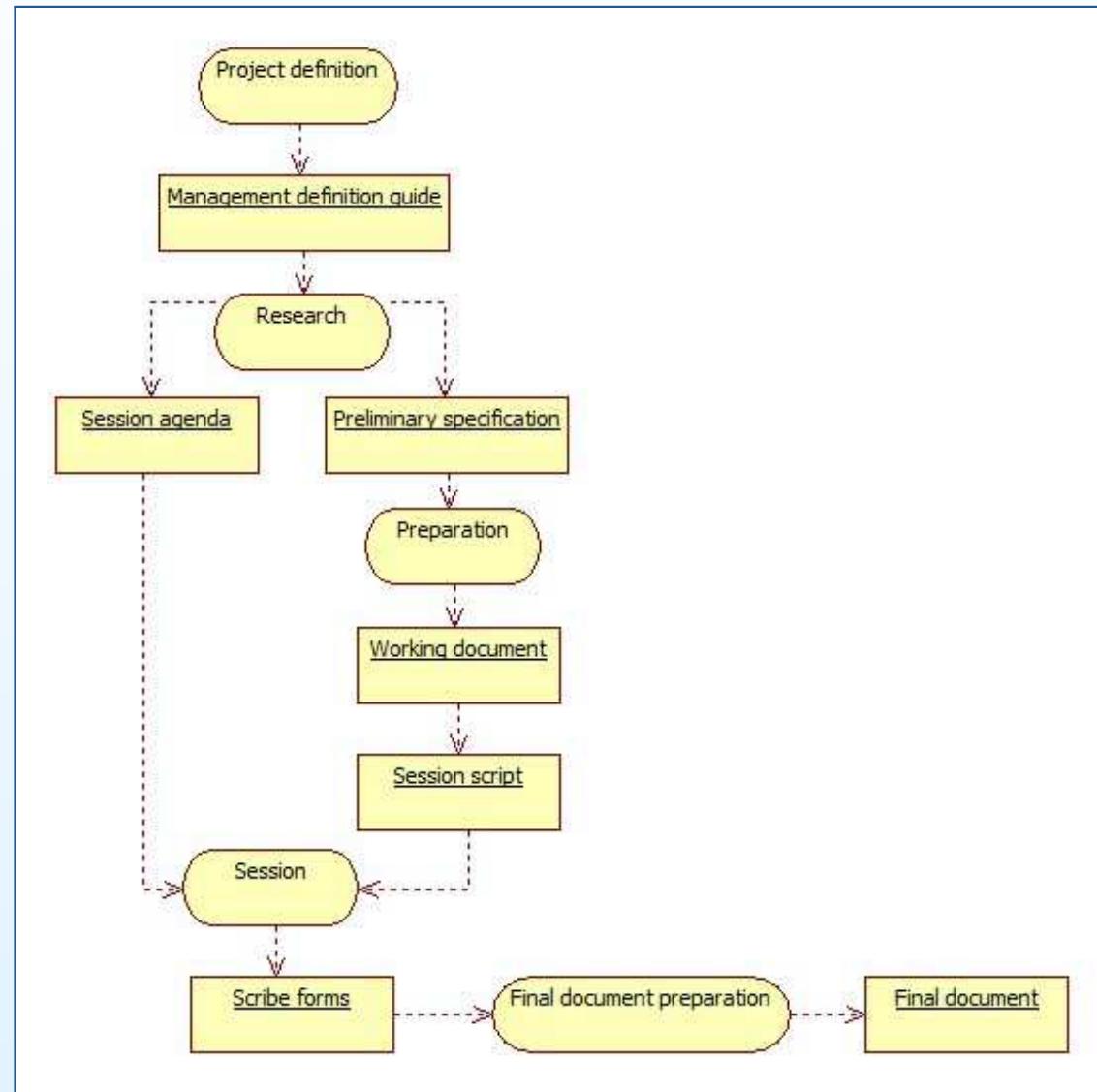
## Metoda JAD

---

- *JAD (Joint Application Design [WoodSilver 1989])* este o metodă de colectare a cerințelor dezvoltată la IBM la sfârșitul anilor '70
- Eficacitatea sa provine din faptul că activitatea de colectare a cerințelor se desfășoară într-o singură sesiune workshop, cu participarea tuturor celor interesați în dezvoltarea sistemului (clienti, utilizatori, dezvoltatori), precum și a unui moderator de sesiune specializat
- Produsul final al workshopului (documentul final JAD) reprezintă specificația completă a sistemului
- Fiind un document redactat de comun acord cu toți participanții, specificația JAD minimizează riscul unor modificări ulterioare ale cerințelor în procesul de dezvoltare
- Succesul unei sesiuni JAD depinde, de cele mai multe ori, de calificarea moderatorului

# Metoda JAD (cont.)

- Activități JAD



# Activități JAD

- *Project definition*
  - Moderatorul JAD interviewează project managerul și clientul, pentru a identifica scopul și obiectivele proiectului, informațiile fiind reținute în *Management Definition Guide*
- *Research*
  - Moderatorul JAD interviewează utilizatorii sistemului, colectează informații relativ la domeniul problemei și elaborează un prim set de cazuri de utilizare. De asemenea, deschide o listă de probleme ce se doresc a fi abordate în cadrul workshop-ului. Rezultatele acestei activități sunt reprezentate de *Session agenda* - agenda de lucru și *Preliminary specification* - specificația preliminară a sistemului
- *Preparation*
  - Moderatorul JAD pregătește sesiunea de lucru, creează o primă versiune a documentului final JAD - *Working document* și compune o echipă constând din project manager, client, utilizatori și dezvoltatori selectați.

# Activități JAD

- *Session*

- Moderatorul JAD coordonează echipa în elaborarea specificației sistemului. O sesiune JAD durează 3-5 zile. Se definesc scenariile, cazurile de utilizare și prototipurile interfeței cu utilizatorul. Toate deciziile sunt documentate (=> *Scribe forms*).

- *Final document preparation*

- Moderatorul pregătește documentul final - *Final document*, revizuind documentul de lucru pentru a include toate deciziile luate în sesiune. Documentul final reprezintă specificația completă sistemului, aprobată în cadrul sesiunii. Acesta este distribuit participanților pentru inspectare. Urmează o sesiune de 1-2 ore în care participanții dezbat modificările și finalizează documentul.

## Referinte

---

- [WoodSilver, 1989] J. Wood, D. Silver, *Joint Application Design*, Wiley, New York, 1989.
- [Grady 1992] R. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, NJ, 1992.
- [IEEE Std. 610.12-1990] Institute of Electrical and Electronics Engineers, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, New York, NY, 1990.
- Use Case Analysis demos

Lesson1, <https://www.youtube.com/watch?v=KeMgPqLCkuo&t=19s>

Lesson2, <https://www.youtube.com/watch?v=8qyHeqInnFU>

Lesson3, <https://www.youtube.com/watch?v=iex3QbSo61c>

Lesson4, <https://www.youtube.com/watch?v=3OWNQddbrOs>

Lesson5, <https://www.youtube.com/watch?v=FrdCX9Gcc4g>